Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Timotej Osolin

Zasnova translacije bioloških modelov v SBML zapis

DIPLOMSKO DELO
UNIVERZITETNI ŠTUDIJSKI PROGRAM PRVE STOPNJE
RAČUNALNIŠTVO IN INFORMATIKA

doc. dr. Miha Moškon
MENTOR

prof. dr. Miha Mraz
SOMENTOR

Ljubljana, 2016

Tematika naloge:

Kandidat naj se v svojem delu osredotoči na objektno orientiran pristop k modeliranju bioloških sistemov z uporabo knjižnice SysBio in programskega jezika Modelica. Dodatno naj preuči predstavitev bioloških modelov v SBML zapisu in možnosti pretvorbe v ta zapis. Razvije naj programsko orodje, ki bo omogočalo pretvobo Modelica modelov razvitih z uporabo knjižnice SysBio v SBML zapis. Pravilnost razvitega orodja naj preveri s pretvorbo izbranih modelov in njihovim uvozom v eno izmed bolj razširjenih orodij s SBML podporo.

IZJAVA O AVTORSTVU DIPLOMSKEGA DELA

Spodaj podpisani izjavljam, da sem avtor dela, da slednje ne vsebuje materiala, ki bi ga kdorkoli predhodno že objavil ali oddal v obravnavo za pridobitev naziva na univerzi ali drugem visokošolskem zavodu, razen v primerih kjer so navedeni viri.
S svojim podpisom zagotavljam, da:

- sem delo izdelal samostojno pod mentorstvom doc. dr. Mihe Moškona in somentorstvom prof. dr. Mihe Mraza,
- so elektronska oblika dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko in
- soglašam z javno objavo elektronske oblike dela v zbirki "Dela FRI".

— Timotej Osolin, Ljubljana, september 2016.

Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Timotej Osolin

Zasnova translacije bioloških modelov v SBML zapis

POVZETEK

Sistemska biologija, kot pomemben člen v verigi številnih področij pri raziskovanju v medicini, uporablja poleg ostalih tehnik tudi kreiranje in urejanje raznovrstnih bioloških modelov. Pri tem tudi izkorišča prednosti objektno orientiranega modeliranja in programskih orodij, namenjenih izdelavi takih modelov. Na Univerzi v Ljubljani so tako razvili knjižnico, imenovano SysBio, kot pomoč pri ustvarjanju bioloških modelov. Za izdelavo knjižnice je bil uporabljen objektno orientiran Modelica programski jezik, ki pa je na področju sistemske biologije in njej sorodnih ved sorazmerno redek. Tako je za kompatibilnost z ostalimi, pogosteje uporabljenimi orodji, potrebna pretvorba v zapis, ki ga ta orodja podpirajo. Poznamo več različnih oblik zapisa, toda v sistemski biologiji je najbolj razširjen zapis SBML (angl. *Systems Biology Markup Language*). V pričujočem delu predstavimo razvoj programa, ki omogoča translacijo iz jezika Modelica v zapis SBML. Translacijo demonstriramo na enostavnem modelu metabolne poti, ki jo uspešno pretvorimo iz Modelica v SBML zapis.

Ključne besede: sistemska biologija, SysBio, SBML zapis, Modelica, translacija, biološki modeli

University of Ljubljana
Faculty of Computer and Information Science

Timotej Osolin

Translation of biologic models to SBML format

ABSTRACT

Systems biology is one of the important fields of study that is helping medicine with its breakthroughs. Amongst other techniques it uses object oriented modelling and corresponding software tools for creation of such models. University of Ljubljana developed a library, named SysBio, as a vital utility for developing biological models. The library was developed using object oriented Modelica programming language, which is relatively rare in the field of systems biology and its related fields. Therefore a translation to a more known format is necessary to ensure the compatibility with the most popular development tools in this field. There are many different types of formats, however SBML (*Systems Biology Markup Language*) is the most widespread format in systems biology. In the following work we present the development of software, which enables the translation from Modelica language to a SBML format. We demonstrate such translation on a simple model of metabolic pathway, which is successfully converted from Modelica to SBML format.

Key words: systems biology, SysBio, SBML format, Modelica, translation, biologic models

ZAHVALA

Zahvaljujem se svojima mentorjema, doc. dr. Mihi Moškonu ter prof. dr. Mihi Mrazu za vso pomoč pri izdelavi diplomskega dela. Zahvaljujem se tudi svoji družini in prijateljem za vso podporo tekom študija.

— Timotej Osolin, Ljubljana, september 2016.

KAZALO

1 Uvod

Na področju medicine se v zadnjih desetletjih pojavljajo odkritja, ki premikajo meje mogočega. Odkrivajo se nove zdravilne učinkovine, ki se nato razvijejo v zdravila, raziskujejo in uspešno preučujejo se številni patogeni, ki so še pred nekaj desetletji veljali za smrtonosne itd. Posledično se povečuje tudi povprečna življenjska doba ljudi povečuje. Na Japonskem je v letu 2015 tako znašala že kar 84 let [1].

T.i. zahodna medicina je v preteklosti rešila že mnogo življenj - in ta trend se le še povečuje. Toda kaj stoji za medicino? Osebni zdravniki, zdravniki specialisti ter ostalo medicinsko osebje se neposredno ne ukvarja z raziskovalnim delom. Njihova naloga je zdraviti ljudi. Premalokrat pa se vprašamo, kdo je tisti, ki odkrije vzroke ter posledice neke bolezni. Oziroma povedano drugače, kako in zakaj je neka snov ali proces v našem telesu škodljiva. Poleg ostalih konvencionalnih področij, kot so farmacija, medicina, biologija, kemoinformatika itd., poznamo tudi dve relativno novi področji, ki drastično povečujeta napredek v medicini in se ju dotaknemo tudi v našem delu. To sta sintezna biologija ter sistemska biologija.

Sintezna biologija preučuje številne biološke sisteme na principu redukcionizma. To

pomeni, da poskuša posamezen sistem (lahko tudi problem) razdeliti na več manjših sistemov, kateri pa so lažje obvadljivi in nato te manjše sisteme, potem ko pozna njihovo delovanje in zgradbo, povezati nazaj v celoto. Tako se raziskovanje poglablja na nižje, bolj enostavne elemente bioloških sistemov. Kot dober primer lahko vzamemo genski inženiring, ki je zelo sorodno področje sintezne biologije [2].

Na drugi strani imamo sistemsko biologijo. Ta se ukvarja z enakimi oziroma podobnimi problemi, toda deluje po principu holizma. Ta narekuje, da je potrebno problem reševati kot celoto. Teorija narekuje, da z izoliranjem posameznih segmentov izgubimo vpogled v delovanje celotnega sistema. Delovanje sistema je namreč drugačno od vsote njegovih elementov. Primer obravnave sistema kot celote so človeški organi - vsak organ je nek sistem, ki ga obravnamo kot celoto [3].

Sistemska biologija je pogost člen v raziskovanju delovanja človeka. Tako so tudi na Univerzi v Ljubljani razvili knjižnico za izdelavo bioloških modelov [4]. Ker pa knižnjica deluje v zelo specifičnem okolju (tj. okolju za modeliranje z objektno orientiranim jezikom Modelica), ki je večini znanstvenikov s področja ved o življenju tuje, jo je potrebno prevesti v zapis, ki ga podpirajo druga orodja za delo s sistemsko biologijo.

1.1 Cilji

Glavni cilj diplomske naloge je izdelava translatorja med že obstoječo knjižnico (poglavje 2) in zapisom SBML (angl. *Systems Biology Markup Language*, poglavje 3). Rezultat translacije je SBML datoteka, ki vsebuje standardiziran zapis za uporabo v orodjih sistemske biologije. Pri tem se bomo omejili predvsem na okvirno zasnovo, podrobno implementacijo delovanja jezika Modelica [5] pa zavoljo enostavnosti izpustili. V kolikor bo okvirna zasnova v zapisu SBML enaka tisti v jeziku Modelica, bomo translacijo zaključili kot uspešno.

1.2 Metodologija

Diplomska naloga je predvsem raziskovalnega tipa - imamo problem (želimo standardiziran zapis), ki pa ga je potrebno raziskati ter nato poiskati ustrezne rešitve. Lahko rečemo, da gre za pilotni primer, saj te knjižnice ni prevedel v standardiziran zapis še nihče. Sicer poznamo nekaj primerov uspešne translacije iz jezika Modelica v zapis SBML [6], toda iz drugih knjižnic, kar pa za naše potrebe ni dovolj. Tako bomo v diplomski nalogi upora-

bili različne pristope - faze. Prva faza je raziskovanje in odkrivanje možnih že obstoječih podobnih rešitev. Naslednja faza je priprava tehničnih zahtev za samo rešitev problema, zadnja faza pa dejanska implementacija rešitve.

1.3 Kratek povzetek

V pričujočih poglavjih smiselno opisujemo sam problem ter podamo predlog za rešitev tega problema. Začnemo s poglavjem 2, ki podrobneje opisuje samo knjižnico, katere translacijo želimo opraviti. Nato opišemo zapis SBML, ki je zapis, v katerega želimo pretvoriti naš model (poglavje 3). V poglavju 4 predstavimo dejanski produkt - program, ki pretvori poljuben model, sestavljen z SysBio knjižnico, v osnovno funkcijsko ekvivalenten model v SBML zapisu. Vse skupaj povzamemo in strnemo v zaključku v poglavju 5.

2 Knjižnica SysBio

V pričujočem poglavju opišemo in analiziramo knjižnico objektov SysBio avtorja prof. dr. Aleša Beliča, ki je bila za namene modeliranja in simulacije na področju sistemske biologije razvita na Univerzi v Ljubljani [7]. Praktičen primer uporabe knjižnice je modeliranje razgradnje holesterola v jetrih [8]. Knjižnica je napisana v modelirnem jeziku Modelica [9]. Ker pa v sistemski biologiji postaja jezik SBML *de facto* standard zapisa biološkega modela, je zaželena splošna pretvorba modela napisanega v jeziku Modelica v ta zapis. Kljub temu, da je na spletu mnogo rešitev za pretvorbo iz različnih programskih jezikov v zapis SBML, neke splošne rešitve za naš problem ni.

Za uspešno pretvorbo je potrebnih več korakov. Prvi je dobro poznavanje knjižnice, katero želimo pretvoriti, seveda pa potrebujemo tudi vsaj osnovno znanje SBML jezika, zaželeno s primeri. Naslednji korak je iskanje podobnosti z že obstoječimi rešitvami, nato pa se lahko lotimo začetnih korakov translacije v želeno obliko.

2.1 Klasifikacija objektov knjižnice

Knjižnica je sestavljena iz cca. 50 objektov/gradnikov, ki predstavljajo osnovne biološke entitete, kot je npr. encim, metabolit in encimska reakcija. Z njihovim povezovanjem enostavno gradimo kompleknejše objekte. Glavna prednost, hkrati pa tudi omejitev te knjižnice, je predpostavljeno normalizirano stabilno stanje sistema. Torej opazujemo le začetno ter končno stanje po perturbaciji, vmesne spremembe (prehodne pojave) pa zanemarimo. To precej poenostavi samo gradnjo modelov in njihove simulacije, saj občutno zmanjša količino kinetičnih parametrov, ki jih moramo pri vzpostavitvi modelov poznati [4]. V tem poglavju bomo opisali osem najpomembnejših in najpogosteje uporabljenih gradnikov knjižnice. Teh osem objektov bomo nato vzeli kot referenčni primer za kasnejšo pretvorbo v SBML zapis.

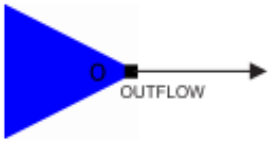
Izvorna koda SysBio knjižnice, ki jo preučujemo, je namenoma nekoliko okrnjena - v izvorni kodi se sicer nahajajo tudi podatki o vizualnem izgledu objektov za lažjo gradnjo modelov v različnih simulacijskih orodjih, ki podpira jezik Modelica (npr. OpenModelica). Teh za vzpostavitev SBML opisa ne potrebujemo. Ker je vizualizacijskih podatkov v primerjavi s funkcionalno kodo zelo veliko, smo jih v naslednjih izsekih kode izpustili, saj kot taki tudi ne vplivajo na sam rezultat izvedbe modela. Enačbe in parametri vseh gradnikov, opisanih v nadaljevanju ter vizualizacija modelov so povzeti po [10].

2.1.1 Gradnik CMass

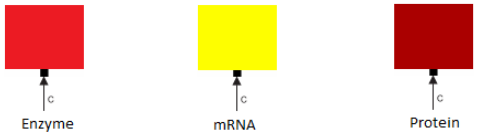
```
within SysBio;
connector Cmass "Transportation of mass"
  Real Q;
  flow Real fi;
end Cmass;
```

Izsek kode 2.1 Izvorna koda **Cmass** objekta.

CMass je najosnovnejši gradnik/objekt v knjižnici SysBio. V osnovi je to konektor, ki povezuje dva objekta. Med dvema konektorjema se lahko nato vzpostavi povezava, preko nje pa poteka izmenjava snovi. Ta pretok (v literaturi se pojavlja tudi izraz $fluks$) označimo kot fi (v nadaljevanju Φ). Koncentracijo snovi na tem konektorju (oziroma v modelu, kateremu ta konektor pripada), označujemo s simbolom Q. Za medsebojno povezane konektorje $C_1, C_2, ..., C_n$ Modelica v ozadju vzpostavi sistema enačb $\sum_{i=1}^{n} \Phi c_i = 0$



Slika 2.1 Grafična predstavitev objekta **Source**.



Slika 2.2 Grafična predstavitev objektov **Enzyme**, **mRNA**, **Protein**.

in $Qc_1 = Qc_2 = ... = Qc_n$. Kot bomo videli v sledečih poglavjih, je **CMass** uporabljen v večini drugih osnovnih objektov za povezavo z drugimi objekti, ki jih bomo v tem poglavju opisali. Funkcionalno kodo gradnika **CMass** prikazuje izsek kode 2.1.

2.1.2 Gradnik Source

Source, vizualiziran na sliki 2.1, si lahko predstavljamo kot vir substrata. Objekt omogoča perturbacijo (t.i. spremenjeno stanje) ob določenem času, po katerem izhodni pretok spremeni svojo vrednost. Gradnik **Source** ima le en konektor, $OUTFLOW$, ki predstavlja izhodni pretok substrata. Pri tem potrebuje tri parametre:

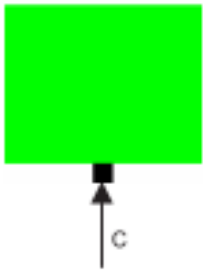
- $massflow_1$: pretok snovi pred perturbacijo - privzeta vrednost 1,
- $massflow_2$: pretok snovi po perturbaciji - privzeta vrednost 1,
- $switchtime$: čas perturbacije - privzeta vrednost 1.

Gradnik opisuje le ena enačba (2.1). Zavoljo enostavnosti je ni potrebno posebej komentirati.

$$OUTFLOW.\Phi = \begin{cases} -massflow_1; & \text{if } t \leq switchtime \\ -massflow_2; & \text{else} \end{cases} \tag{2.1}$$

2.1.3 Gradniki Enzyme, Protein, mRNA

Objekti **Enzyme**, **Protein** in **mRNA** predstavljajo istoimenske objekte - torej encim, protein ter mRNA. Vse te gradnike obravnavamo hkrati, saj imajo povsem enake parametre, spremenljivke in enačbe ter se razlikujejo le po namembnosti. Pri tem **mRNA** (angl.



Slika 2.3 Grafična predstavitev objekta **Metabolite**.

messenger RNA) predstavlja mRNA molekule kot vmesni produkt pri prepisovanju gena v protein. Vsi omenjeni objekti imajo v sebi neko koncentracijo, ta pa se skozi čas lahko spreminja. Spremembe objektov so prožene s strani drugih objektov, ki le-tega bodisi porabljajo ali proizvajajo. Q_0 je začetna koncentracija objektov (privzeta vrednost 1) in je parameter, Q pa koncentracija encima, realizirana kot spremenljivka. Poleg tega objekti vsebujejo tudi razgradnjo (proteina, encima oz. mRNA), ki je sorazmeren s konstanto razgradnje k (del enačbe (2.3)). Tako gradniki vsebujejo dva koeficienta za degradacijo, k_1 in k_2. Njuni privzeti vrednosti sta $k_1 = 0.01$ ter $k_2 = 0$. Objekt omogoča tudi perturbacijo razgradnje, ki se zgodi ob času *switchtime*, ob katerem konstanta k_2 dobi neko drugo vrednost. S parametrom *switchtime* lahko spreminjamo čas perturbacije za degradacijo. Privzeta vrednost parametra *switchtime* je 0, kar pomeni, da perturbacije ni. Enačbe (2.2), (2.3), (2.4) veljajo tako za objekt **Enzyme**, objekt **Protein** kot tudi za objekt **mRNA**. Vsi gradniki so predstavljeni na sliki 2.2.

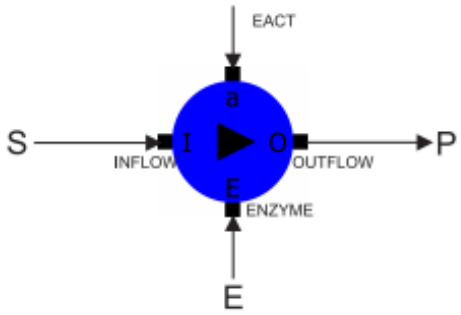
$$k = \begin{cases} k_1; & \text{if } t \leq switchtime \\ k_1 * \dfrac{1}{1+k_2}; & \text{else} \end{cases}, \tag{2.2}$$

$$\frac{dQ}{dt} = \begin{cases} 0; & \text{if } Q \leq 0 \text{ and } C.\Phi - k * Q < 0 \\ C.\Phi - k * Q; & \text{else} \end{cases}, \tag{2.3}$$

$$C.Q = Q. \tag{2.4}$$

2.1.4 Gradnik Metabolite

Produkt metabolizma je v knjižnici predstavljen z gradnikom **Metabolite**. Vizualiziran je na sliki 2.3. Njegova koncentracija se spreminja na podlagi enačbe (2.5). Od gradnikov



Slika 2.4 Grafična predstavitev objekta **ESReaction**.

predstavljenih v razdelku 2.1.3 se razlikuje po tem, da ne vsebuje enačbe za degradacijo. Ima le en konektor, in sicer C. Spremenljivka Q opisuje koncentracijo metabolita, parameter Q_0 pa njegovo začetno koncentracijo (privzeta vrednost je 1). Enačba (2.5) opisuje delovanje gradnika **Metabolite**.

$$\frac{dQ}{dt} = \begin{cases} 0; & \text{if } Q \leq 0 \text{ and } C.\Phi < 0 \\ C.\Phi; & \text{else} \end{cases} \tag{2.5}$$

2.1.5 Gradnik ESReaction

ESReaction opisuje encimsko reakcijo, ki sledi Michaelis-Menten kinetiki. Seveda za pravilno interpretacijo reakcije potrebujemo vhodne podatke o pretoku (**Cmass**), podatke o encimu, ki v reakciji sodeluje (**Enzyme**) ter parametre za reakcijo. Ti parametri tudi določajo, za kakšno reakcijo gre in kakšen bo njen rezultat.

Poleg spremenljivke Q, ki opisuje koncentracijo kompleksa znotraj gradnika, ima gradnik **ESReaction** zaradi svoje kompleksnosti tudi več konektorjev za povezavo z drugimi gradniki (konektorji so tudi predstavljeni na sliki 2.4):

- $INFLOW$: vhodni pretok substrata (S),
- $ENZYME$: vhodni pretok encima (E),
- $OUTFLOW$: izhod produkta (P),
- $EACT$: regulator aktivnosti encima (opcijsko).

Pri tem pa jih definirajo sledeči parametri:

- k_C: hitrost formacije kompleksa,

- k_P: hitrost formacije produkta,
- r: reverzibilnost reakcije - privzeta vrednost 0.01,
- $Q_0 = w$: začetna koncentracija kompleksa,
- M: število molekul substrata, porabljenih v eni reakciji - privzeta vrednost 1.

V kolikor imamo konektor $EACT$ priključen, se aktivnost encima določa po enačbi (2.6). Kot izhod nato dobimo nek produkt, ki pa je drugačen od začetnega stanja encima (spremenila ga je ravno ta reakcija). Porabo substrata opisuje enačba (2.7), tvorjenje produkta pa enačba (2.8). Spremembo v koncentraciji kompleksa opisuje enačba (2.9). Končna poraba encima pri reakciji je enaka spremembi kompleksa, kar opisuje enačba (2.10).

$$e_act = \begin{cases} EACT.Q; & \text{if EACT exists} \\ 1; & \text{else} \end{cases}, \tag{2.6}$$

$$S.FI = INFLOW.\Phi = M * (e_act * k_C * S.Q^M * E.Q - r * k_P * Q), \tag{2.7}$$

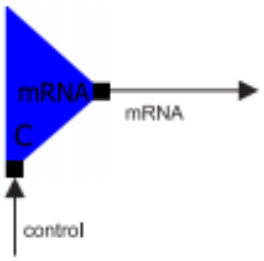
$$P.FI = -OUTFLOW.\Phi = k_P * Q - e_act * r * k_C * P.Q * E.Q, \tag{2.8}$$

$$\frac{dQ}{dt} = S.FI - P.FI = INFLOW.\Phi + OUTFLOW.\Phi, \tag{2.9}$$

$$E.FI = ENZYME.\Phi = \frac{dQ}{dt} = S.FI - P.FI. \tag{2.10}$$

2.1.6 Gradnik GeneExpressionControl_lin

GeneExpressionControl_lin je model (linearnega) genskega izražanja. Poenostavljeno povedano, mRNA zapis (opisan v razdelku 2.1.3) nosi neko genetsko informacijo, ta pa se lahko naprej izraža kot sprememba v drugih objektih (npr. encimu) po enačbi (2.11). Vsebuje dva konektorja, $mRNA$, ki opisuje mRNA izhodni pretok ter *control*, konektor, s katerim lahko nadzorujemo izhodni pretok mRNA. Ta konektor je opcijski. Edini parameter, ki opisuje gradnik **GeneExpressionControl_lin** je k, ki določa hitrost izražanja mRNA. Njegova privzeta vrednost je 0.00001. Grafično je gradnik v okolju OpenModelica predstavljen na sliki 2.5.



Slika 2.5 Grafična predstavitev objekta **GeneExpressionControl_lin**.

Slika 2.6 Grafična predstavitev objekta **EFormation_lin**.

$$m\text{RNA}.\Phi = \begin{cases} -k * control.Q; & \text{if } control \text{ is connected} \\ -k; & \text{else} \end{cases} \tag{2.11}$$

2.1.7 Gradnik EFormation_lin

EFormation_lin je linearna reakcija, ki opisuje translacijo mRNA v protein. Vhodni pretok encima se tako na podlagi podanega koeficienta spremeni in preusmeri na izhodni pretok. Tako se mRNA sprosti in prepiše gen v protein po enačbi (2.13), toda pri tem mRNA ostane nespremenjen (se ne porabi), kar vidimo v enačbi (2.12). **EFormation_lin** ima dva konektorja: $INFLOW$, ki opisuje mRNA vhodni pretok ter $OUTFLOW$, ki opisuje izhodni pretok proteina. Za pretvorbeno konstanto uporablja parameter k, ki ima privzeto vrednost 0.01. Grafična predstavitev se nahaja na sliki 2.6.

$$INFLOW.\Phi = 0, \tag{2.12}$$

$$outflow.\phi = -k * inflow.Q. \tag{2.13}$$

```
model test
  SysBio.Source source1(massflow2 = 0.1, switchtime = 500);
  SysBio.Metabolite metabolite1;
  SysBio.ESReaction eSReaction1;
  SysBio.Enzyme enzyme1;
  SysBio.mRNA mRNA1;
  SysBio.EFormation_lin eFormation_lin1;
  SysBio.Metabolite metabolite2;
  SysBio.NElimination nElimination1;
equation
  connect(metabolite2.C, nElimination1.inflow);
  connect(eSReaction1.outflow, metabolite2.C);
  connect(eFormation_lin1.outflow, enzyme1.C);
  connect(mRNA1.C, eFormation_lin1.inflow);
  connect(eSReaction1.enzyme, enzyme1.C);
  connect(metabolite1.C, eSReaction1.inflow);
  connect(source1.out, metabolite1.C);
end test;
```

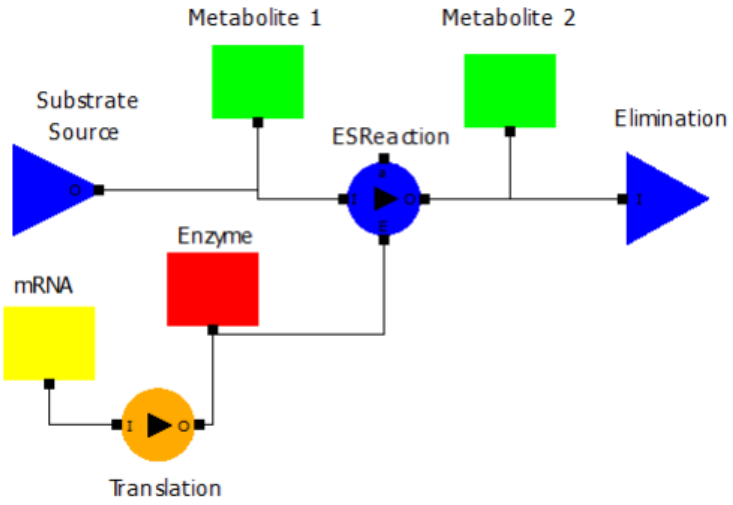
Izsek kode 2.2 Primer modela SysBio knjižnice

2.2 Primeri uporabe knjižnice

V pričujočem poglavju predstavimo tri primere knjižnice. Začeli bomo z najbolj osnovnim, teoretičnim metabolnim modelom, ki ga lahko razširimo tudi na bolj kompleksne modele. Drugi in tretji primer pa sta modela realnih sistemov, razvita v raziskovalnih skupinah na Univerzi v Ljubljani. Prvi izmed takih modelov je model sinteze holesterola [8], drugi pa model hepatocita v povezavi z okoliškimi tkivi in organi, poimenovan tudi SteatoNet. Vsi modeli in njihove izvorne kode so dostopni na [4].

2.2.1 Enostaven metabolni model

Izsek kode 2.2 prikazuje primer uporabe knjižnice SysBio. V nadaljevanju razdelka predstavimo model osnovnega segmenta v metabolnem omrežju. S širjenjem takega segmenta bi lahko enostavno prišli do modela kompleksnejšega sistema, kot so npr. jetrne celice (hepatociti) v povezavi z okoliškimi organi in tkivi (glej sliko 2.9). Enostaven model, na katerem bomo tudi demonstrirali pretvorbo v SBML, je sestavljen iz osmih gradnikov ter sedmih povezav med njimi. Povezave med posameznimi gradniki so smiselno postavljene, npr. gradnik **Enzyme** je povezan z gradnikom **EFormation_lin** kot njegov izhod, gra-



Slika 2.7 Grafična predstavitev modela enostavne metabolne poti v orodju OpenModelica [11].

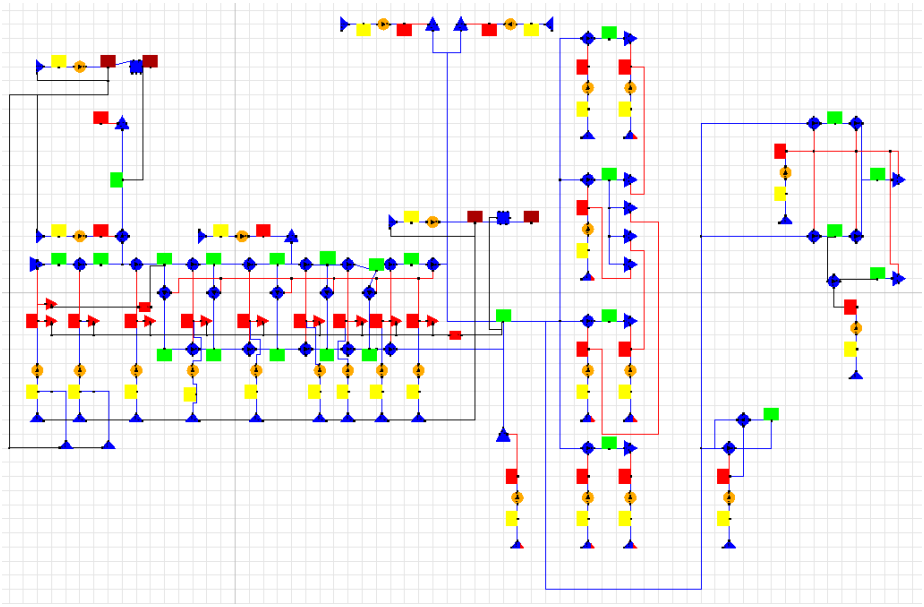
dnik **mRNA** pa kot njegov vhod. Podrobnejša razlaga o smiselni povezavi gradnikov je v razdelku 2.1 ali v viru [4]. Zgoraj opisani enostaven model, vizualiziran v orodju OpenModelica, prikazuje slika 2.7.

2.2.2 Model sinteze holesterola

Eden izmed modelov realnih bioloških sistemov, razvitih s pomočjo knjižnice SysBio, je tudi model sinteze holesterola. Povišana raven holesterola predstavlja znan faktor tveganja pri razvoju srčnožilnih obolenj. Razloge za njegovo povišanje lahko identificiramo tudi s pomočjo matematičnega modeliranja. V ta namen je bil razvit model sinteze holesterola v jetrih, ki ga prikazuje slika 2.8. Model predstavlja prvo uspešno aplikacijo knjižnice SysBio. Kompleksnost tega modela je precej večja kot pri enostavnem teoretičnem modelu. To je posledica precej večjega števila uporabljenih gradnikov in povezav. Vpelje se tudi nekaj novih gradnikov, ki jih v pričujočem delu nismo opisali, smo pa jih upoštevali pri translaciji tega modela v poglavju 4.4.2. Podrobnejši opis teh gradnikov je opisan v izvorni kodi knjižnice SysBio [4].

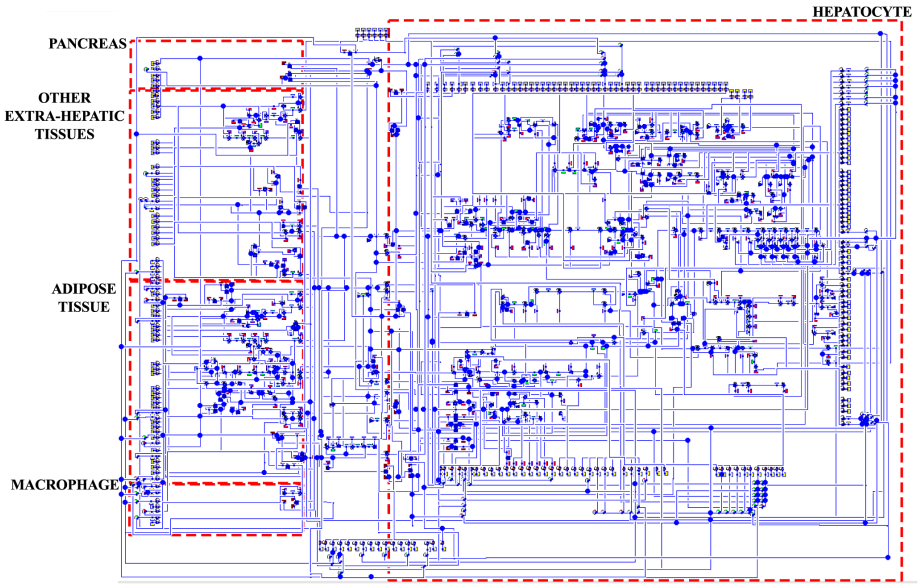
2.2.3 SteatoNet

SteatoNet predstavlja model jetrne celice (hepatocita) v povezavi okoliškimi organi in tkivi, ki je bil razvit za vpogled v nastanek kompleksnih bolezni kot je nealkoholna



Slika 2.8 Grafična predstavitev modela sinteze holesterola v orodju OpenModelica.

zamaščenost jeter (angl. *Non-Alcoholic Fatty Liver Disease, NAFLD*)[4]. Slika 2.9 prikazuje model SteatoNet. Takoj opazimo visoko kompleksnost, pri pregledu izvorne kode pa še večje število dodatnih gradnikov.



Slika 2.9 Grafična predstavitev modela SteatoNet v orodju OpenModelica [12].

3 SBML zapis

SBML (angl. *Systems Biology Markup Language*) predstavlja specializirano različico XML jezika [13]. Torej je tako kot njegov sorodnik XML sorazmerno enostaven za obdelavo na računalniku. Kot njegovo alternativo lahko omenimo SBGN (angl. *Systems Biology Graphical Notation*), ki ponuja grafični vmesnik za lažjo človeško obdelavo podatkov. SBML se uporablja predvsem na področju sistemske biologije, na kar namiguje že njegovo ime, podpira pa ga že veliko programskih orodij s tega področja. Avtorji jezika definirajo SBML kot nek vmesni člen med drugimi jeziki. Tako predstavlja povezovalni jezik, ki bi bil vsaj v osnovnih funkcijah povsem interoperabilen s preostalimi modelirnimi jeziki za modeliranje v sistemski biologiji in bi omogočal izmenljivost modelov med različnimi orodji in raziskovalnimi skupinami.

3.1 Nivoji jezika

V splošnem poznamo tri nivoje zapisa SBML. Za njih velja, da je prehod iz nižjega na višji nivo vedno možen, iz višjega na nižji nivo pa ni zagotovljen. Za vsako novejšo različico je potrebna tudi posodobitev programske opreme, ki uporablja zapis SBML. Ravno to

je tudi eden izmed glavnih razlogov, zakaj so nivoji zasnovani tako, da drug drugega ne izključujejo. To pomeni, da so stare različice zapisa še vedno uporabne in obstajajo v sožitju, četudi avtorji izdajo novejšo različico. Tako avtorji programske opreme, ki v svojem razvoju uporabljajo jezik SBML, ne potrebujejo nujno takojšnje posodobitve na najnovejšo različico, čeprav jim to avtorji zapisa SBML močno priporočajo [13].

Med pomembnejše novosti, vpeljane v najnovejši različici zapisa SBML, lahko štejemo podporo za sestavljanje hiearhičnih modelov, podporo za t.i. modele v ravnovesnem stanju (angl. *steady-state*), vpeljavo skupin itd. Ker se v knjižnici, ki jo obdelujemo, uporablja med drugim tudi ravnovesna stanja, bomo za našo pretvorbo uporabili nivo 3 (najnovejši) SBML zapisa. Primer zapisa SBML prikazuje izsek kode 3.1.

```
<math xmlns="http://www.w3.org/1998/Math/MathML"
 xmlns:sbml="http://www.sbml.org/sbml/level3/version1/core">
 <apply>
  <plus/>
   <ci> k </ci>
   <apply>
    <csymbol encoding="text" definitionURL=
    "http://www.sbml.org/sbml/symbols/delay" />
    <ci> x </ci>
    <cn sbml:units="second"> 0.1 </cn>
   </apply>
 </apply>
</math>
```

Izsek kode 3.1 Primer SBML zapisa za enačbo $k + delay(x, 0.1)$[14].

3.2 Uporaba SBML pri pretvorbi

V tem poglavju bomo opisali osnovne gradnike SBML zapisa, ki jih bomo največ uporabljali. Opisi so povzeti po [15].

- *Compartment*: Razdelek (angl. *compartment*) je eden najosnovnejših gradnikov. V njega lahko shranjujemo končno število zvrsti (angl. *species*). Model lahko vsebuje več razdelkov, a v našem primeru bomo uporabili le enega.
- *Species*: Kot zvrst obravnavamo entitete enakega tipa, ki se nahajajo v istem razdelku. Sicer lahko kot zvrst obravnavamo vsako entiteto, ki ima v kontekstu

modela nek smisel, a v našem primeru bodo zvrsti kar gradniki, opisani v razdelku 2.1.

- *Reaction*: Reakcija (angl. *reaction*) opisuje transformacijo ali proces med zvrstmi. Ta lahko spremeni obstoječe zvrsti, generira nove ali pa jih v svojem procesu tudi porabi. Pri nas bomo reakcije opisali z enosmernimi in dvosmernimi povezavami med gradniki. Knjižnica SysBio namreč že vsebuje reakcije, ki jih želimo uporabiti. Vsaka reakcija v SBML mora vsebovati reaktante (angl. *reactants*) ter produkte (angl. *products*), ki pa se lahko spreminjajo tekom reakcije. V dvosmernih reakcijah reaktanti nastopajo tudi kot produkti in obratno.

- $listOfSpecies$: Vse entitete enakega tipa moramo v SBML zapisu združevati. To storimo s seznami - v našem primeru bomo potrebovali štiri. Prvi je *listOfSpecies*, ki združuje zvrsti. Znotraj njega nato poljubno definiramo zvrsti, ki jih nato uporabljamo v reakcijah.

- $listOfReactions$: Drugi seznam je *listOfReactions*. Ta združuje reakcije - znotraj njega definiramo reakcije. Te reakcije uporabljajo prej definirane zvrsti. Ker v SBML zapisu velja hiearhični princip, znotraj seznama reakcij obravnavamo reakcije, znotraj teh pa lahko definiramo sezname reaktantov ter produktov.

- $listOfReactants$: Naslednji seznam nam definira reaktante v reakciji. En reaktant je načeloma ena zvrst - tako moramo referencirati prej definirane zvrsti s pomočjo ključne besede *speciesReference*. Seznam reaktantov je tako seznam zvrsti, ki se v reakciji porabljajo.

- $listOfProducts$: Podobno kot pri seznamu reakcij, tudi pri seznamu produktov nastopajo že prej definirane zvrsti. Edina razlika je, da zvrsti v tem seznamu nastanejo oziroma spremenijo vrednost glede na reakcijo, ki jih spreminja.

3.3 Podpora jezika SBML v orodjih za modeliranje

Na uradni spletni strani jezika SBML [13] najdemo zbirko mnogih orodij, ki podpirajo SBML notacijo. V nadaljevanju na kratko opišemo nekatera izbrana orodja.

3.3.1 Pregled orodij

V zbirki na uradni SBML strani se trenutno nahaja že več kot 250 orodij (september 2016) [13]. Med seboj se razlikujejo v različnih funkcionalnostih, ki jih podpirajo. Nekateri tako omogočajo le vizualni pregled zapisa SBML ter ustvarjanje modelov, drugi pa popolno simulacijo bioloških modelov od začetka do konca, poleg tega pa še statistično in analitično podporo z dodatnimi funkcionalnostmi.

Orodja podpirajo tudi druge programske jezike, s pomočjo katerih nato enostavneje gradimo nove modele ali pa podporne programe za SBML zapis. Pri izbiri programa je tako potrebno biti pozoren tudi na podprte programske jezike. Mnogi podpirajo bolj splošno znane, kot so npr. *Java*, *C++*, *Python*, *Mathematica*, itd., nekateri pa bolj specializirane, npr. *SPARQL*. Pomemben faktor pri izbiri je tudi, ali programska oprema podpira uvoz in izvoz SBML zapisa. Ta namreč ni samoumeven, saj iz drugih programskih jezikov (kot npr. v našem primeru jezik Modelica) pretvorba v zapis SBML ni vedno podprta. Zbirka vseh programskih orodij, po kateri smo tudi povzeli pričujoči razdelek, se nahaja na [16].

3.3.2 Izbor najbolj ustreznih orodij

Med poplavo orodij za izdelavo modelov s pomočjo zapisa SBML se je morda včasih težko odločiti, kateri je za nas najbolj primeren. Odločili smo se izbrati nekaj takih, za katere smatramo, da so med uporabniki razširjeni ali pa imajo potencial za nadaljni razvoj. Med njimi najdemo tako prosto dostopna orodja kot tudi plačljiva.

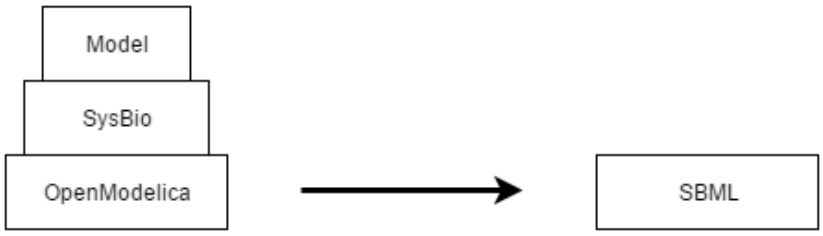
Prvi, ki ga velja omeniti in ga uporabimo tudi pri preverjanju uspešnosti pretvorbe naših modelov, je orodje COPASI. To orodje podpira kreairanje novih modelov, njihovo simulacijo ter analizo tako pridobljenih podatkov. Na voljo je na vseh popularnejših platformah (Windows, Linux, Mac OS X) ter je brezplačen tako za akademsko kot tudi komercialno uporabo. Za preverjanje naših modelov smo ga izbrali zaradi njegove enostavnosti uporabe ter hitro in predvsem pregledno avtomatsko vizualizacijo. Orodje podpira tudi razvoj lastnih vtičnikov, tako da bi lahko naš program v prihodnosti vanj tudi vgradili.

Med izborom za testiranje na našem primeru smo med kandidati imeli tudi orodje TinkerCell. Orodje tako kot COPASI podpira kreairanje, simulacijo ter analizo modelov v zapisu SBML. Uporablja druge programske jezike za podporo zapisu SBML ter je ravno tako kot COPASI prosto dostopen za akademsko in komercialno rabo. Zelo znano

orodje je tudi orodje Wofram SystemModeler, ki ga je razvil avtor popularnega orodja za reševanje matematičnih problemov Mathematica. Njegovi glavni pomankljivosti sta tudi odsotnost na tržišču za Linux sisteme in odsotnost brezplačne različice.

4 Izvedba translacije

Za namen translacije iz jezika Modelica v zapis SBML smo razvili program, ki poljuben model, kreiran s pomočjo SysBio knjižnice, pretvori v SBML zapis. Ta je ekvivaleten zapisu v jeziku Modelica v strukturi modela, ne pa tudi po samih enačbah, ki se uporabljajo v ozadju modela. Program smo razvili s pomočjo orodja Visual Studio (2015) in sicer s pomočjo Windows Forms objektov v samem orodju ter v jeziku C#. V sledečih razdelkih opisujemo potek same translacije, kot je sprogramirana v našem translatorju. Sama pretvorba se zgodi v treh korakih - branje vhodne datoteke (.mo), shranjevanje v začasne podatkovne strukture ter pisanje v novo datoteko (.xml). Slika 4.1 prikazuje



Slika 4.1 Slikovna predstavitev pretvorbe modela iz jezika OpenModelica v zapis SBML.

diagram translacije.

4.1 Podatkovne strukture

Prvi korak k uspešni translaciji je pravilno strukturiranje začasnih podatkovnih struktur, ki jih nato uporabimo v sami pretvorbi. Tu smo definirali nove C# razrede (angl. *class*). En razred tako definira en gradnik, opisan v poglavju 2. Ker za uspešno translacijo smatramo že pravilno pretvorbo same strukture modela, se posledično poenostavi tudi definicija posameznega razreda. Vsak razred ima enega ali več atributov, ki se nato uporabljajo za povezovanje razredov in so ekvivalentni konektorju (glej razdelek 2.1.1). Vsi atributi pripadajo podatkovnemu tipu *String*. Skupno vsem razredom je, da imajo atribut *name* - vsak gradnik ima namreč med drugim tudi svoje unikatno ime, ki služi kot identifikator, s pomočjo katerega nato povezujemo gradnike med seboj.

Nekateri gradniki imajo le zgoraj omenjeni atribut - to so pripadniki vrste (glej podpoglavje 3.2). Ti namreč nimajo dodatnih operacij, ki bi vplivale na sam gradnik. Gradniki **Metabolite** (2.1.4), **mRNA** (2.1.3) in **Enzyme** (2.1.3) pripadajo vrstam in tako nimajo dodatnih atributov, imajo pa tudi reakcijski del - razgradnjo.

Če smo tri zgoraj omenjene gradnike klasificirali kot vrste v zapisu SBML, pa vse ostale gradnike razvrstimo kot reakcije. To pomeni, da imajo poleg atributa *name* tudi vsaj en drug atribut, ki opisuje izhodno ali vhodno povezavo. Ti atributi nam tako povedo, katere vrste nastopajo kot reaktanti in produkti reakcije (glej razdelek 3.2). Če konektorje posameznega gradnika definiramo s pomočjo atributov, dobimo sledeče dodatne atribute za vsak razred (na levi strani je zapisano ime razreda, na desni pa dodatni atributi):

- *Source* - outflow,
- *GeneExpressionControl_lin* - mRNA,
- *Elimination* - inflow,
- *EFormation_lin* - inflow, outflow,
- *ESReaction* - inflow, outflow, enzyme.

Kot opazimo, opcijskih konektorjev, opisanih v poglavju 2.1, v naših podatkovnih strukturah zaradi poenostavitve modela nismo upoštevali in so tako ignorirani. Kot primer podatkovne strukture izsek kode 4.1 prikazuje razred **ESReaction**.

```
public class ESReaction{
      public String name { get; set; }
      public String inflow { get; set; }
      public String outflow { get; set; }
      public String enzyme { get; set; }
  }
```

Izsek kode 4.1 Funkcionalna koda za razred ESReaction.

4.2 Razčlenjevalnik

Vsak model, sestavljen s pomočjo knjižnice SysBio, ima določeno osnovno strukturo. Razumevanje te nam pomaga pri branju ter razčlenjevanju (angl. parsing) vhodne datoteke - modela. Osnovno strukturo modela prikazuje izsek kode 4.2. Pri tem je potrebno poudariti, da je omenjeni izsek kode le psevdokoda, v kateri so namenoma izpuščeni segmenti kode, ki niso relevantni za našo translacijo.

```
model imeModela
    //Deklaracija gradnikov
    SysBio.gradnik imeGradnika1
    SysBio.gradnik imeGradnika2
equation
    //Deklaracija povezav gradnikov
    connect(imeGradnika1.konektor1,imeGradnika2.konektor2)
end imeModela;
```

Izsek kode 4.2 Psevdokoda modela knjižnice SysBio.

Kot vidimo v izseku kode 4.2, se vsak model začne z imenom modela, ki sledi ključni besedi *model*. Zatem sledi deklaracija gradnikov - vsak gradnik je definiran kot *SysBio.gradnik*, pri čemer *gradnik* zamenjamo z poljubnim gradnikom, opisanim v razdelku 2.1. Zatem sledi še ime tega gradnika - ta se kasneje zapiše v atribut *Name*, omenjen v razdelku 4.1. Ključna beseda *equation* nam pove, da je deklaracij gradnikov konec in začnejo se naštevati povezave gradnikov med seboj. Pri tem vrstni red povezav ni pomemben. Dva gradnika povežemo med seboj s pomočjo ključne besede *connect*, ki jo lahko tretiramo kot funkcijo. Ta sprejme dva parametra, in sicer konektor prvega gradnika ter konektor drugega gradnika, ki ju želimo povezati (konektorji so opisani pri gradnikih v razdelku 2.1). Model se zaključi s ključno besedo *end*, ki ji sledi ime modela

- ta je seveda enak imenu modela pri začetni deklaraciji. Praktičen in enostaven primer modela prikazuje izsek kode 2.2, dostopen na [4].

Programska realizacija razčlenjevanja modela je, ko enkrat poznamo sestavo modela, enostavna. Začnemo z deklaracijo dinamičnih seznamov vsakega razreda posebej. To nam kasneje omogoča enostavno sprotno dodajanje novih gradnikov, ki jih beremo iz datoteke. Po začetnih deklaracijah seznamov ter samega modela, začnemo s samim branjem. Naša implementacija bere vhodno datoteko vrstico po vrstico. Nato vsako ključno besedo (*model*, *SysBio.gradnik*, *connect*, ...), ki jo prebere, obravnava posebej. Tako npr. ključno besedo *SysBio.Source* obravnava kot gradnik *Source* ter kreira istoimensko strukturo z imenom *imeGradnika*, to strukturo (razred) pa nato doda v že prej deklariran dimanični seznam. To nato iterira skozi vse vrstice vhodne datoteke, rezultat tega pa so napolnjeni seznami razredov - gradnikov ter njihove povezave.

4.3 Pretvorba v SBML zapis

Ko imamo na voljo sezname razredov, je pretvorba teh v zapis SBML enostavna. V naši implementaciji smo še nekoliko poenostavili sam zapis v datoteko z namenom zmanjšanja števila vhodno/izhodnih operacij nad datotekami. Tako najprej zapišemo celotno SBML datoteko v podatkovno strukturo *String*, nato pa to enkratno zapišemo v novo datoteko XML.

Zapis SBML je v osnovi zastavljen tako, da je priporočljivo poleg definicije vsake vrste zapisati vsaj tudi njeno začetno koncentracijo oz. t.i. začetno stanje. Čeprav ta parameter od SBML nivoja 3 naprej ni obvezen [17], nas program za delo z SBML zapisom na to opozori in sam model tudi popravi, v kolikor bi pri simulaciji prišlo do napake. Kljub temu, da naš produkt v tej fazi ne upošteva oz. uporablja principa simuliranja dinamike, pa nastavi začetno koncentracijo vseh produktov na neko začetno vrednost (v našem primeru je ta vrednost 1.0).

Vsaka XML datoteka ima, tako kot Modelica datoteka, določeno strukturo. Opis te strukture se nahaja v poglavju 3. Tako se moramo tudi pri našem zapisu v izhodno datoteko držati te strukture - začnemo z deklaracijo XML ter SBML podatkov. Nato začnemo z opisom modela, ki ima enako ime kot ime modela v prebrani Modelica datoteki. Sledijo zapisi *compartment*, *listOfSpecies* ter *listOfReactions* (vsi opisani v razdelku 3.2). Pri zapisih *listOfSpecies* iteriramo skozi sezname razredov, ki opisujejo vrste, torej

Metabolite, **mRNA**, **Enzyme**, pri zapisih *listOfReactions* pa skozi ostale sezname razredov. Pri tem dodamo vsak razred v nov zapis. Najlažje je potek translacije opisati skozi primer. Izsek kode 4.3 prikazuje vse vrste, ki nastopajo v Modelica datoteki, opisani v izseku kode 2.2. Tako vidimo, da vsako vrsto zapišemo v novo vrstico, ime vrste pa je enako atributu *name* posameznega razreda.

```
<listOfSpecies>
    <species id="mRNA1" name="mRNA1" />
    <species id="metabolite1" name="metabolite1" />
    <species id="metabolite2" name="metabolite2" />
    <species id="enzyme1" name="enzyme1" />
</listOfSpecies>
```

Izsek kode 4.3 Del SBML datoteke, ki opisuje vrste.

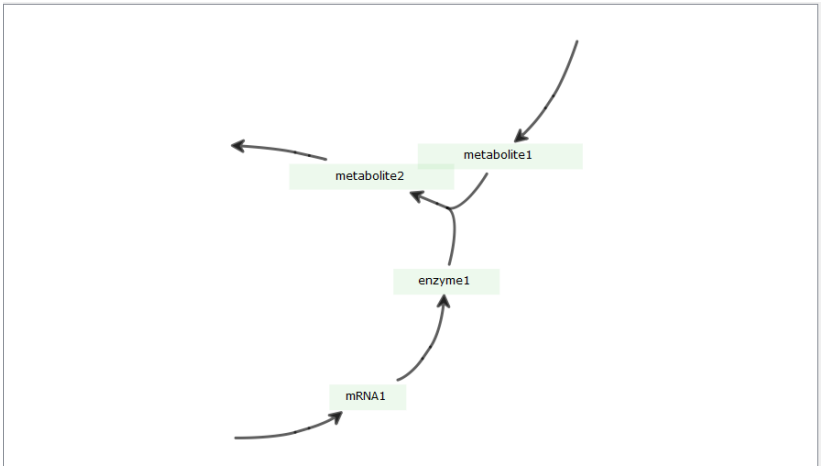
Po definiciji zvrsti so na vrsti vse reakcije - povezave, ki nastopajo v modelu. Izsek kode 4.4 prikazuje eno reakcijo - *source*, ki je opisana v izseku kode 2.2. Ime reakcije je enako atributu razreda, ki opisuje to reakcijo, vse vhodne povezave naštejemo v seznamu *listOfReactants*, vse izhodne pa v *listOfProducts*. Ker že iz samih definicij gradnikov vemo, katere povezave so vhodne, katere pa izhodne, je zapis reakcij trivialen.

```
<reaction id="source1" name="source1" reversible="false">
    <listOfReactants>
    </listOfReactants>
    <listOfProducts>
        <speciesReference species="metabolite1"/>
    </listOfProducts>
</reaction>
```

Izsek kode 4.4 Del SBML datoteke, ki opisuje eno izmed reakcij.

4.4 Zgledi pretvorbe

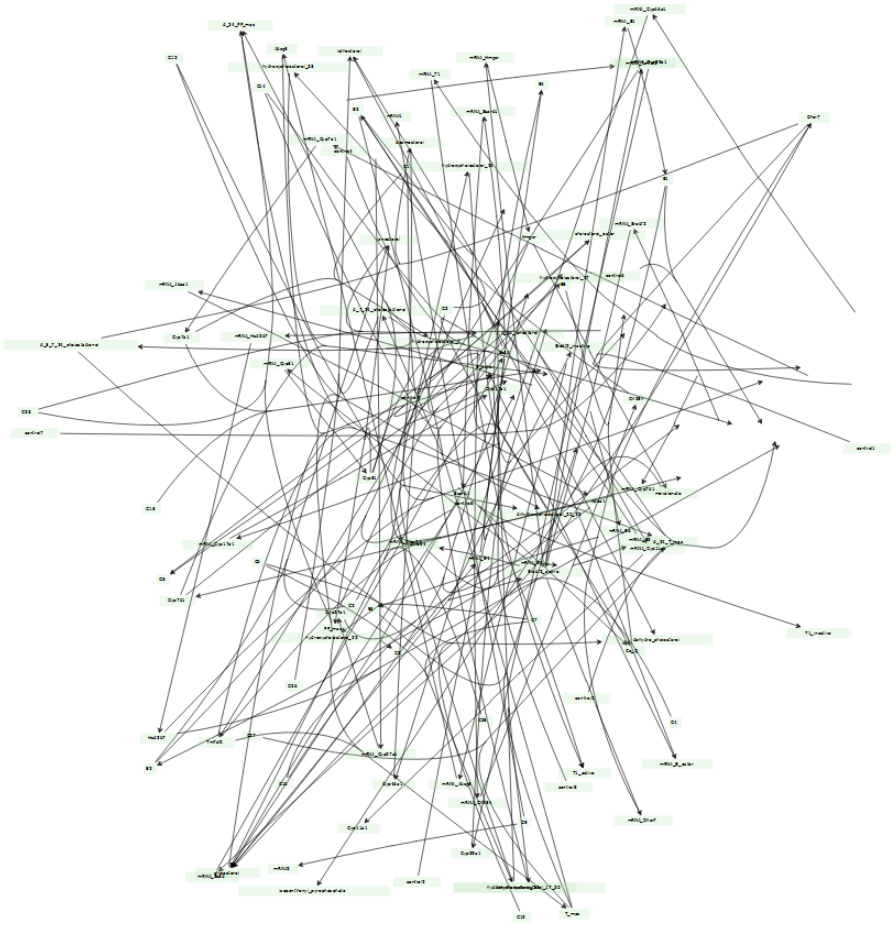
Pretvorbo bomo demonstrirali na dveh zgledih predstavljenih v razdelku 2.2, in sicer na zgledu enostavne metabolne poti (razdelek 2.2.1) in na modelu sinteze holesterola v jetrih (razdelek 2.2.2). Uspešnost pretvorbe bomo testirali z uvozom SBML zapisa v orodje COPASI in z grafičnim prikazom uvoženega modela.



Slika 4.2 Grafična predstavitev osnovnega modela v orodju COPASI.

4.4.1 Enostaven metabolni model

Kot zgled vzemimo model iz razdelka 2.2.1. Tako naš program najprej prebere celotno datoteko (izsek kode 2.2) in jo shrani v začasno spremenljivko. Nato začne z obdelavo vrstico po vrstico. Začetni korak je inicializacija modela z imenom test. Zatem se berejo posamezni gradniki (našteti po vrstnem redu) – **Source**, **Metabolite**, **ESReaction**, **Enzyme**, **mRNA**, **EFormation_lin**, **Metabolite** in **NElimination** ter pripadajoča imena. V tem primeru sicer opazimo, da je vsak gradnik razen **Metabolite** uporabljen le enkrat. To je seveda izjema, vsak gradnik se lahko v modelu pojavi poljubnokrat, a vsakič mora biti inicializiran z novim imenom. Po deklaraciji vseh gradnikov ter njihovo shrambo v posamezne dinamične sezname, program nadaljuje z razčlenjevanjem preko ključne besede connect. Začnejo se tvoriti povezave med posameznimi gradniki. Tako se, skladno z Modelica datoteko, polnijo posamezne povezave v gradnikih. Možno je seveda, da kakšna povezava ostane nepovezana. V tem primeru program enostavno povezavo ustvari v zrak oz. povedano drugače, smatrajo se, da izvirajo ali ponikujejo izven opazovanega sistema. Po uspešnem razčlenjevanju sledi zapis v datoteko. Rezultat je tako datoteka .xml po SBML standardu (glej izsek kode 4.5). To datoteko smo nato uspešno odprli s pomočjo orodja COPASI. Grafično predstavitev pretvorjene datoteke prikazuje slika 4.2.



Slika 4.3 Grafična predstavitev modela sinteze holesterola v orodju COPASI.

4.4.2 Sinteza holesterola

Naslednji model, ki ga želimo pretvoriti v zapis SBML je model sinteze holesterola. Postopek pretvorbe je enak kot v razdelku 4.4.1. Za potrebe pretvorbe tega modela smo v program dodali še dodatne gradnike, ki niso opisani v razdelku 2.1. Uspešnost pretvorbe modela prikazuje slika 4.3. Modeliranje v orodju COPASI se ponavadi načrtuje od začetka do konca. Tako pri modeliranju ročno spreminjamo postavitev posameznih gradnikov. Ker pa je naš program prevedel že obstoječi model, COPASI pa nima integriranih obstoječih metod za pravilno oz. bolj smiselno pozicioniranje posameznih gradnikov, dobimo za rezultat navidez kaotično sliko. Pravilnost pretvorbe tako potrjujemo s tem, da COPASI uspešno odpre našo datoteko in jo zna tudi interpretirati.

```
<?xml version="1.0" encoding="UTF-8"?>

<sbml xmlns="http://www.sbml.org/sbml/level2/version3" level="2">
    <model id="test" name="model_test">
        <listOfCompartments>
            <compartment id="compartment" name="compartment"/>
        </listOfCompartments>
        <listOfSpecies>
            <species id="mRNA1" name="mRNA1" />
            <species id="metabolite1" name="metabolite1" />
            <species id="metabolite2" name="metabolite2" />
            <species id="enzyme1" name="enzyme1" />
        </listOfSpecies>
        <listOfReactions>
            <reaction id="source1" name="source1" reversible="false">
                <listOfReactants>
                </listOfReactants>
                <listOfProducts>
                    <speciesReference species="metabolite1"/>
                </listOfProducts>
            </reaction>
            <reaction id="geneExpressionControl_lin1" name="gec_lin1" >
                <listOfReactants>
                </listOfReactants>
                <listOfProducts>
                    <speciesReference species="mRNA1"/>
                </listOfProducts>
            </reaction>
            <reaction id="eFormation_lin1" name="eFormation_lin1" >
                <listOfReactants>
                    <speciesReference species="mRNA1"/>
                </listOfReactants>
                <listOfProducts>
                    <speciesReference species="enzyme1"/>
                </listOfProducts>
            </reaction>
            <reaction id="eSReaction1" name="eSReaction1" >
                <listOfReactants>
                    <speciesReference species="metabolite1"/>
                    <speciesReference species="enzyme1"/>
                </listOfReactants>
```

```
                <listOfProducts>
                    <speciesReference species="metabolite2"/>
                </listOfProducts>
            </reaction>
            <reaction id="nElimination1" name="nElimination1" >
                <listOfReactants>
                    <speciesReference species="metabolite2"/>
                </listOfReactants>
                <listOfProducts>
                </listOfProducts>
            </reaction>
        </listOfReactions>
    </model>
</sbml>
```

Izsek kode 4.5 Poenostavljena pretvorjena SBML datoteka, izhod našega programa.

5 Zaključek

V delu smo predstavili zasnovo translacije modelov, predstavljenih v jeziku Modelica in z uporabo knjižnice SysBio, v zapis SBML. Uspešnost pretvorbe s pomočjo razvitega programa smo demonstrirali na dveh testnih primerih, tj. fiktivni metabolni poti in modelu sinteze holesterola v jetrih. S tem smo izpolnili cilj pridobitve osnovne izmenljivosti že obstoječe knjižnice z bolj uveljavljenimi orodji. Tako smo lahko s pomočjo orodij za obdelavo modelov v SBML zapisu (v našem primeru smo uporabili COPASI) grafično predstavili model, ki smo ga pred tem skonstruirali v jeziku Modelica. Pri primerjavi obeh grafičnih predstavitev hitro opazimo, da je kompatibilnost standardiziranega zapisa zelo pomembna. Isti model namreč lahko predstavimo na množico različnih načinov.

Ker smo se v delu odločili omejiti število funkcionalnosti, ki jih naš translator podpira (raziskovali in načrtovali smo namreč dokaj neraziskano področje), je v našem delu še nekaj prostora za izboljšave. Na prvem mestu tu velja omeniti simuliranje dinamike. Ena izmed poglavitnih prednosti jezika Modelica je namreč tudi spremljanje sprememb parametrov in spremenljivk modelov skozi čas. Izpeljava in pretvorba take dinamike v SBML zapis je časovno precej potratna, saj je potrebno vse možne primere obravnavati indivi-

dualno. Uvedba dinamike je tako primerna za naslednji korak v razvoju izpopolnjenega translatorja.

V delu smo obravnavali osnovni segment metabolnega omrežja. Z uvedbo množice takšnih elementov lahko s sorazmerno enostavnostjo zgradimo modele precej kompleksnejših segmentov, z razumevanjem zgradbe človeškega telesa pa tudi npr. celotne človeške organe. Tako bi lahko v nadaljnjem raziskovanju uporabili še zapletenejše modele. Te bi sicer bilo treba pred tem še zgraditi v okolju Modelica. Kompleksnost gradnje takih modelov je precej velika, hkrati pa je za razvoj potrebno tudi poglobljeno znanje biologije, bioinformatike in sorodnih področij. Naš program je tako dobra osnova za pretvorbo kompleksnejših in naprednejših modelov v prihodnosti. Z implementacijo našega programa tako upamo, da bomo vsaj nekoliko poenostavili sodelovanje raziskovalne skupine, ki uporablja knjižnico SysBio, z ostalimi raziskovalnimi skupinami na področju medicine in na področju ved o življenju.

Literatura

[1] World Health Organization-life expectancy, http://www.who.int/gho/mortality_burden_disease/life_tables/situation_trends/en/, Zadnji dostop: 6.6.2016.

[2] Synthetic Biology, http://syntheticbiology.org/, Zadnji dostop: 1.9.2016.

[3] Institute for Systems Biology, https://www.systemsbiology.org/, Zadnji dostop: 1.9.2016.

[4] SysBio Modelica library, http://lrss.fri.uni-lj.si/bio/sysbio/index.html, Zadnji dostop: 31.5.2016.

[5] Modelica and the Modelica association, https://www.modelica.org/, Zadnji dostop: 6.6.2016.

[6] J. Brugard, D. Hedberg, M. Cascante, G. Cedersund, A. Gomez-Garrido, D. Maier, E. Nyman, V. Selivanov, P. Stralfors, Creating a bridge between modelica and the systems biology community, Proceedings 7th Modelica Conference, Como, Italy, Sep. 20-22 (2009) 473–479.

[7] Knjižnica SysBio, https://bio.tools/tool/ELIXIR-SI-hackathon/SysBio%20modelica%20library/1, Zadnji dostop: 25.5.2016.

[8] A. Belič, J. Ačimovič, A. Naik, M. Goličnik, Analysis of the steady-state relations and control-algorithm characterisation in a mathematical model of cholesterol biosynthesis, Simulation Modelling Practice and Theory 33 (2013) 18–27.

[9] P. Fritzon, Introduction to Object-Oriented Modeling and Simulation with Modelica using OpenModelica, https://www.openmodelica.org/images/docs/tutorials/modelicatutorialfritzson.pdf, Zadnji dostop: 5.7.2016.

[10] M. Moškon, T. Cvitanović, D. Rozman, M. Mraz, Notes on the derivation and brief documentation of SysBio library, http://lrss.fri.uni-lj.si/bio/sysbio/files/sysbio.pdf, Zadnji dostop: 28.5.2016.

[11] M. Moškon, T. Cvitanović, M. Mraz, Understanding Complex Diseases with Object Oriented Modelling , http://lrss.fri.uni-lj.si/bio/sysbio/files/IBD4Health_OOM.pdf, Zadnji dostop: 3.9.2016.

[12] A. Naik, D. Rozman, A. Belič, The first integrated human metabolic model with multi layered regulation to investigate liver associated pathologies, PLOS Computational Biology.

[13] System Biology Markup Language, http://sbml.org/Main_Page, Zadnji dostop: 17.5.2016.

[14] M. Hucka, F. Bergmann, S. Hoops, S. Keating, S. Sahle, J. Schaff, L. Smith, D. Wilkinson, The systems biology markup language (sbml): Language specification for level 3 version 1 core, Nature Precedings (2010) 22–24.

[15] The SBML: Language Specification for Level 3 Version 1 Core, http://sbml.org/Documents/Specifications, Zadnji dostop: 31.5.2016.

[16] SBML - The Systems Biology Markup Language, http://sbml.org/SBML_Software_Guide/SBML_Software_Matrix, Zadnji dostop: 3.9.2016.

[17] SBML - The Systems Biology Markup Language, http://sbml.org/Community/Wiki/SBML_Level_3_Core/No_default_values, Zadnji dostop: 3.9.2016.